Matrix Inversion by Rank Annihilation

By David J. Edelblute

1. Remarks. The problem of matrix inversion has been extensively explored and a number of methods have been found to solve this problem. However, no one "best" method has been found and the computer programmer must still choose a method which is suited to his particular needs.

The purpose of this paper is to outline a method which allows one to bring the matrix into memory one column at a time and overlap input time with computing time. Since the original matrix need not be stored in memory, the use of memory is also efficient, and the number of computations is as small as any known to the writer. If the original matrix is already in memory, this method can also be used to avoid destroying the original matrix.

2. Theory. The reader may easily verify that if A is a nonsingular matrix, U and V are column vectors, and $(A + UV^T)$ is nonsingular, then

$$(A + UV^T)^{-1} = A^{-1} - \frac{(A^{-1}U)(V^TA^{-1})}{1 + V^TA^{-1}U}. \tag{1}$$

Householder remarks that if $1 + V^TA^{-1}U = 0$ then the matrix $(A + UV^T)$ is singular [2].

The repeated use of (1) to find the inverse of an $n \times n$ matrix, B, is known as the method of rank annihilation. To use this method we write

$$B = D + \sum_{i=1}^{n} U_iV_i^T, \tag{2}$$

where D is a matrix of known inverse. Thus we can define a sequence of matrices $\{C_i\}$ such that

$$\begin{aligned} C_0 &= D, \\ C_k &= D + \sum_{i=1}^{k} U_iV_i^T, \\ &= C_{k-1} + U_kV_k^T, \end{aligned} \qquad k = 1, \cdots, n.$$

If all C_i are nonsingular formula (1) gives a sequence of matrices $\{C_i^{-1}\}$, and $C_n^{-1} = B^{-1}$.

Clearly the expansion (2) is not unique. Thus we have the problem of choosing a sequence of U_i and V_i which will produce B^{-1} with the fewest computations and which will require a minimum amount of memory space in a computer. No optimum solution is known to the writer. The purpose of this paper is to point out some advantages of one extremely simple expansion.

Let B be the $n \times n$ matrix to be inverted and let the matrix $U = B - I$, where I is the identity matrix. Partition U by columns so that $U = [U_1U_2 \cdots U_n]$. Let V_i be the ith column vector of the identity matrix. Then

$$B = I + \sum_{i=1}^{n} U_iV_i^T.$$

This is the expansion which we shall discuss.

Received June 18, 1965.

3. Comments. The following advantages of this procedure may be observed:

1. The matrices C_i need not be stored in memory. The matrices C_i^{-1} are computed one after another and may all be stored in the same place.

2. The vectors V_i need never be stored.

3. The entire matrix B need not be stored in the memory, since it can be brought in one column at a time. This also allows the computer to overlap reading time and computation time.

Thus C_i^{-1} is the only square matrix which must be stored in memory at one time.

4. The row vector $V_i^T C_{i-1}^{-1}$ need not be computed or stored when computing C_i^{-1}, since it is simply the ith row vector of C_{i-1}^{-1}. Thus, we must store only C_{i-1}^{-1}, the ith column vector of B (with 1 subtracted from the ith element to produce U_i), and the product $C_{i-1}^{-1}U_i$ to compute C_i^{-1}. This requires $n(n+2)$ memory locations. This requirement can be further reduced to $n(n+1)$ by storing $C_{i-1}^{-1}U_i$ in the nth column of C_{i-1}^{-1}, since this column is V_n until the last pass.

5. The product $V_i^T C_{i-1}^{-1}U_i$ requires no computation since $C_{i-1}^{-1}U_i$ is already calculated and $V_i^T(C_{i-1}^{-1}U_i)$ is simply the ith element of $C_{i-1}^{-1}U_i$.

4. Number of Computations. The simple multiplication $C_{i-1}^{-1}U_i$ requires n^2 multiplications. Another $n^2 + n$ multiplications and divisions are necessary to calculate $(C_{i-1}^{-1}U_i)(V_i^T C_{i-1}^{-1})/(1 + V_i^T C_{i-1}^{-1}U_i)$. Further, this operation must be repeated n times. Thus, ignoring computations used in indexing, the number of multiplications and divisions in the process is $2n^3 + n^2$.

The programmer may observe, however, that the last $n - i$ columns of C_i^{-1} contain only zeros, except for ones in the diagonal positions. If one takes advantage of this fact, the number of multiplications and divisions can be cut almost in half. For large matrices, then, the result is approximately $n^3 + \frac{1}{2}n^2$.

5. Disadvantages. It was noted earlier that the method will fail if any of the C_i are singular. This will happen if and only if any of the principal submatrices of B formed by the deletion of the last $n - i$ rows and columns of B are singular. If B is already stored in the memory, this trouble can usually be remedied by changing the order in which the U_i are used. If this is done, however, some of the above advantages are lost.

6. Conclusion. Although it is possible for the method outlined to fail for some nonsingular matrices, this method appears to be quite practical for use on a digital computer. The minimum amount of memory required for storage is $n(n+1)$ words plus the program to invert an $n \times n$ matrix. The number of multiplications and divisions necessary to invert large matrices is approximately $n^3 + \frac{1}{2}n^2$, and computation time can be overlapped with tape reading time.

The writer wishes to thank Dr. S. T. Parker and Dr. J. W. Meux, both of Kansas State University and Dr. R. N. Goss of the U. S. Navy Electronics Laboratory for their assistance with this paper.

7. Example. Let

$$B = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 5 \end{bmatrix}.$$

Then

$$U_1 = \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix}, \quad U_2 = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}, \quad \text{and } U_3 = \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix}.$$

Thus

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \\ 5 \end{bmatrix} [1 \quad 0 \quad 0] + \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} [0 \quad 1 \quad 0] + \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix} [0 \quad 0 \quad 1].$$

$C_0 = I, C_0^{-1} = I.$

$$C_1^{-1} = I - (IU_1)(V_1^T I)/(1 + V_1^T I U_1) = I - \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 4 & 0 & 0 \\ 5 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{5}{2} & 0 & 1 \end{bmatrix}.$$

$$C_2^{-1} = C_1^{-1} - \frac{\left(\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{5}{2} & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}\right)\left([0 \quad 1 \quad 0]\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{5}{2} & 0 & 1 \end{bmatrix}\right)}{1 + [0 \quad 1 \quad 0]\left(\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{5}{2} & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}\right)}.$$

$$C_3^{-1} = C_2^{-1} - (C_2^{-1}U_3)(V_3^T C_2^{-1})/(1 + V_3^T C_2^{-1} U_3) = B^{-1}.$$

U. S. Navy Electronics Laboratory
San Diego, California

1. A. Ralston & H. S. Wilf, *Mathematical Methods for Digital Computers*, Wiley, New York, 1960, pp. 73–77. MR **22** #8680.
2. A. S. Householder, *Principles of Numerical Analysis*, McGraw-Hill, New York, 1953, pp. 79, 83. MR **15**, 470.

Certain Expansions of the Basic Hypergeometric Functions

By Arun Verma

1. Introduction. In a recent paper Jerry L. Fields and Jet Wimp [7] have used a very elegant method by induction through Laplace transform to derive a number of expansions of hypergeometric functions. In this paper, I have used certain basic integrals and the method of induction to derive certain expansions of basic hypergeometric functions of a very general character. The following usual notation has been used throughout the paper. Let

$$[a]_n = [q^a]_n = (1 - q^a)(1 - q^{a+1}) \cdots (1 - q^{a+n-1}), \qquad [a]_0 = 1,$$

and

$$(a + x)_\lambda = \prod_{k=0}^{\infty}\left(\frac{a + xq^k}{a + xq^{k+\lambda}}\right), \qquad (1 - x)_\infty = \prod_{n=0}^{\infty}(1 - xq^n).$$

Received May 24, 1965.